

Data Visualization (part 2)

Journalism 303: Advanced graphing
techniques with ggplot2

Martin Frigaard

2021-09-27

ggplot2 = a grammar for data visualization



Load the packages



```
install.packages("tidyverse")
library(tidyverse)
```

Outline



Recap ggplot2

Graphing preliminaries

- *Data Wrangling*
- *Tidying*

Variable Distributions

- *Histograms, density plots, violin plots*

Line Graphs

Adding Text

- *Annotations, labeling values*

Reference Lines

Advanced Faceting

- *facet_wrap(), facet_wrap_paginate(), facet_geo()*

Resources

Link to slides

<https://mjfrigaard.github.io/csuc-data-journalism/slides.html>

Link to exercises

<https://mjfrigaard.github.io/csuc-data-journalism/lessons-exercises.html>

Recap of **ggplot2**



Recap of `ggplot2`



In the previous lesson, we covered:

1) The grammar of graphics

- `ggplot2` is a language of *layers*, organized linearly
- `ggplot2`'s layers give us a "*linear ordering of phrases*" to build an infinite number of graphs "*which convey a gnarly network of ideas.*"
- "**Infinitely extensible**"

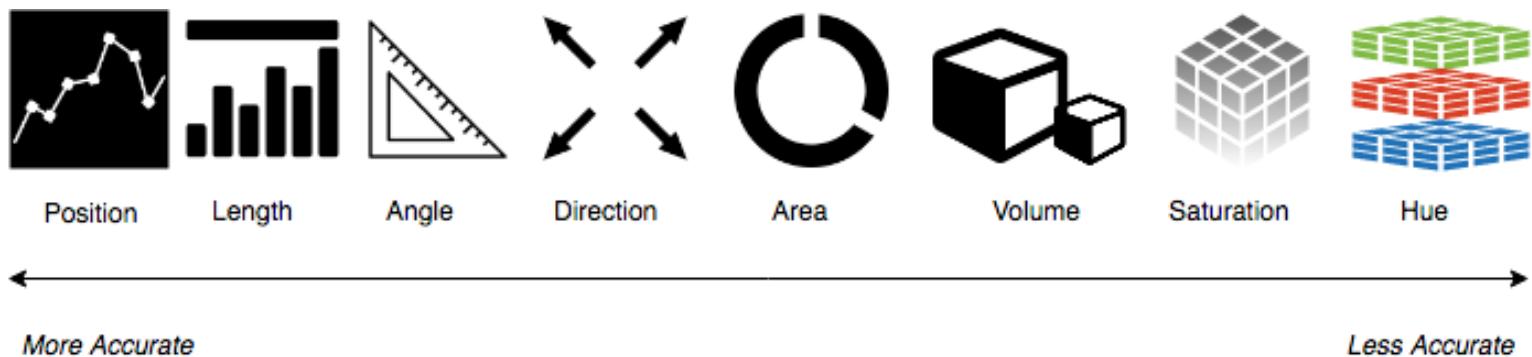
Recap of ggplot2 (cont)



In the previous lesson, we covered:

- 1) The grammar of graphics
- 2) Identifying graph aesthetics

- position (*x* and *y*), size, color, shape, etc.



Recap of `ggplot2` (cont)



In the previous lesson, we covered:

1) The grammar of graphics

2) Identifying graph aesthetics

3) Recognizing and using **geoms**

- Scatter plot = `geom_point()`
- Box plot = `geom_boxplot()`
- Line graph = `geom_line()`
- Bar graph = `geom_histogram()`, `geom_bar()`, `geom_col()`

Recap of `ggplot2` (cont)



In the previous lesson, we covered:

- 1) The grammar of graphics
- 2) Identifying graph aesthetics
- 3) Recognizing and using **geoms**
- 4) Labels and factes (exercises)

- **Build labels first!**
- Facet for subplots of levels in a grouping variable

Before we start...



Things to consider (1)



Recognize the needs of your audience

level of data literacy, subject matter expertise, etc.

Check and communicate data quality with stakeholders

let them know the good and the bad news

Identify the correct data visualization (based on the data)

single variable, bivariate, and multivariate graphs

Things to consider (2)



Incorporate feedback from stakeholders/audience into graphs

ask them to be part of the process

Design visualizations with the appropriate detail and annotations

inform (and do not mislead) the audience

Getting started



1) Clearly define the question or problem

- Start with a general goal, broad question, or novel problem
- Move towards specific tasks

2) Matching the measurements to metrics

- '*Measurements*' are what we care about
- '*Metrics*' are the available data

3) Is a data visualization the best choice?

- Not always...

COVID and Transportation Habits



Example: How has COVID changed our modes of transportation?

What kind of measurements would these be?

how are people traveling (walk, drive, etc.)

What would these data look like?

what would the columns and rows look like?



Apple Mobility Data



Fortunately, these data exist!

Apple released mobility data:

<https://covid19.apple.com/mobility>

Import these data below:

```
AppleMobRaw <- readr::read_csv("https://bit.ly/36tTVpe")
```

*Use **Raw** as a prefix or suffix for data in its most 'raw' state*

Raw Apple Mobility Data



```
AppleMobRaw %>% View("Apple")
```

Dataset Variables

	geo_type	region	transportation_type	sub-region	country	2020-01-13	2020-01-14	2020-01-15	2020-01-16	2020-01-17	2020-01-18	2020-01-19	2020-01-20	2020-01-21	2020-01-22	2020-01-23	2020-01-24
1	country/region	Albania	driving	N/A	N/A	100	95.30	101.43	97.20	103.55	112.67	104.83	94.39	94.07	93.51	92.94	10
2	country/region	Albania	walking	N/A	N/A	100	100.68	98.93	98.46	100.85	100.13	82.13	95.65	97.78	95.39	94.24	9
3	country/region	Argentina	driving	N/A	N/A	100	97.07	102.45	111.21	118.45	124.01	95.44	95.13	95.42	97.66	99.42	11
4	country/region	Argentina	walking	N/A	N/A	100	95.11	101.37	112.67	116.72	114.14	84.54	101.37	106.12	104.91	102.56	10
5	country/region	Australia	driving	N/A	N/A	100	102.98	104.21	108.63	109.08	89.00	99.35	103.53	106.80	107.40	115.65	10
6	country/region	Australia	transit	N/A	N/A	100	101.78	100.64	99.58	98.34	86.97	99.87	107.29	109.13	107.03	106.64	10
7	country/region	Australia	walking	N/A	N/A	100	101.31	101.82	104.52	113.73	100.24	98.57	104.38	108.51	103.88	112.40	11
8	country/region	Austria	driving	N/A	N/A	100	101.14	104.24	112.21	117.23	117.22	105.17	100.70	102.67	104.33	107.89	12
9	country/region	Austria	walking	N/A	N/A	100	101.55	105.59	112.24	123.36	131.05	89.93	100.60	103.30	105.28	110.89	13
10	country/region	Belgium	driving	N/A	N/A	100	101.19	107.49	107.67	117.38	119.32	102.68	103.67	107.83	110.00	108.65	11
11	country/region	Belgium	transit	N/A	N/A	100	98.67	105.01	105.87	113.55	110.73	100.53	105.62	110.39	114.35	111.03	11
12	country/region	Belgium	walking	N/A	N/A	100	101.46	110.44	118.86	139.10	174.99	111.72	105.08	113.37	119.64	118.63	14
13	country/region	Brazil	driving	N/A	N/A	100	99.71	100.90	101.88	113.69	114.38	91.06	97.54	98.14	97.91	98.51	10
14	country/region	Brazil	transit	N/A	N/A	100	102.45	104.28	100.20	97.06	81.01	69.05	100.37	109.18	106.12	102.51	9
15	country/region	Brazil	walking	N/A	N/A	100	106.30	104.75	99.05	104.13	101.18	69.84	97.33	105.36	102.28	95.88	10
16	country/region	Bulgaria	driving	N/A	N/A	100	102.56	104.73	104.35	114.79	118.01	105.42	100.78	99.44	103.52	106.11	11
17	country/region	Bulgaria	walking	N/A	N/A	100	101.90	99.61	100.06	117.97	128.66	99.64	105.30	98.60	104.38	109.35	13
18	country/region	Cambodia	driving	N/A	N/A	100	100.75	99.33	96.00	98.78	103.64	99.16	97.65	96.87	95.13	81.87	8
19	country/region	Cambodia	walking	N/A	N/A	100	98.06	99.70	98.00	106.81	106.84	106.98	102.84	100.70	103.34	100.77	10
20	country/region	Canada	driving	N/A	N/A	100	102.91	99.74	105.17	122.34	102.50	84.46	102.63	104.31	107.37	110.29	12

Wrangling Apple Mobility Data



Wrangling Apple Mobility Data



What variables do we have?

How are these variables formatted?

How do we need to change them?

...with a focus on answering our question

View Apple Mobility Data (head)



```
AppleMobRaw %>% head()
```

geo_type	region	transportation_type	sub-region	country
<chr>	<chr>	<chr>	<chr>	<chr>
country/region	Albania	driving	NA	NA
country/region	Albania	walking	NA	NA
country/region	Argentina	driving	NA	NA
country/region	Argentina	walking	NA	NA
country/region	Australia	driving	NA	NA
country/region	Australia	transit	NA	NA

6 rows | 1-5 of 322 columns

View Apple Mobility Data (tail)



```
AppleMobRaw %>% tail()
```

geo_type	region	transportation_type	sub-region
<chr>	<chr>	<chr>	<chr>
county	York County	walking	South Carolina
county	York County	walking	Pennsylvania
county	Young County	driving	Texas
county	Yuba County	driving	California
county	Yuma County	driving	Arizona
county	Yuma County	walking	Arizona

6 rows | 1-4 of 322 columns

Tidying Apple Mobility Data



Tidy dates and mobility into `date` and `dir_request` ('relative usage of directions')

```
AppleMobRaw %>%
  tidyr::pivot_longer(cols = -c(geo_type:country),
  names_to = "date", values_to = "dir_request")
```

geo_type	region	transportation_type	sub-region	country	
<chr>	<chr>	<chr>	<chr>	<chr>	
country/region	Albania	driving	NA	NA	
country/region	Albania	driving	NA	NA	
country/region	Albania	driving	NA	NA	
country/region	Albania	driving	NA	NA	
country/region	Albania	driving	NA	NA	
country/region	Albania	driving	NA	NA	
country/region	Albania	driving	NA	NA	
country/region	Albania	driving	NA	NA	
country/region	Albania	driving	NA	NA	
country/region	Albania	driving	NA	NA	

1-10 of 60 rows | 1-5 of 7 columns

Previous **1** **2** **3** **4** **5** **6** Next

Manipulate Apple Mobility Data



Remove missing values in `country` and `sub-region` and `clean_names()`

```
AppleMobRaw %>%
  tidyverse::pivot_longer(cols = -c(geo_type:country),
    names_to = "date", values_to = "dir_request") %>%
  # remove missing country data
  dplyr::filter(!is.na(country) & !is.na(`sub-region`)) %>%
  # clean names
  janitor::clean_names() %>% View("TidyApple")
```

New Tidy Apple Mobility Data!



names_to =

values_to =

Screenshot of a data editor showing a table titled "TidyApple". The table has columns: geo_type, region, transportation_type, sub_region, country, date, and dir_request. The "date" and "dir_request" columns are highlighted with green and yellow backgrounds respectively. The data shows mobility trends for Aachen, Germany, from January 13 to February 01, 2020.

	geo_type	region	transportation_type	sub_region	country	date	dir_request
1	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-13	100.00
2	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-14	100.73
3	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-15	102.86
4	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-16	102.65
5	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-17	109.39
6	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-18	109.62
7	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-19	98.21
8	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-20	102.74
9	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-21	103.85
10	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-22	102.01
11	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-23	101.40
12	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-24	107.57
13	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-25	109.13
14	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-26	97.51
15	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-27	101.64
16	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-28	100.59
17	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-29	103.33
18	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-30	104.31
19	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-01-31	113.82
20	city	Aachen	driving	North Rhine-Westphalia	Germany	2020-02-01	113.59

Assign Apple Mobility Data to TidyApple



This looks good, so assign to `TidyApple`

```
AppleMobRaw %>%
  tidyr::pivot_longer(cols = -c(geo_type:country),
  names_to = "date", values_to = "dir_request") %>%
  # remove missing country data
  dplyr::filter(!is.na(country) & !is.na(`sub-region`)) %>%
  # clean names
  janitor::clean_names() -> TidyApple
```

TidyApple: Format Variables



Our **TidyApple** data still needs some attention.

```
glimpse(TidyApple)
```

```
## Rows: 1,055,293
## Columns: 7
## $ geo_type      <chr> "city", "city", "city", "city"...
## $ region        <chr> "Aachen", "Aachen", "Aachen", ...
## $ transportation_type <chr> "driving", "driving", "driving"...
## $ sub_region    <chr> "North Rhine-Westphalia", "Nor...
## $ country       <chr> "Germany", "Germany", "Germany"...
## $ date          <chr> "2020-01-13", "2020-01-14", "2...
## $ dir_request   <dbl> 100.00, 100.73, 102.86, 102.65...
```

The **date** variable needs to be formatted as a **date**

```
TidyApple %>%
  mutate(date = lubridate::ymd(date)) -> TidyApple
```

TidyApple: Rename Variables



Let's rename `transportation_type` to `trans_type`

```
TidyApple %>%
  rename(trans_type = transportation_type) -> TidyApple
```

TidyApple: Check Formatted Variables



Re-check **TidyApple** data.

```
glimpse(TidyApple)
```

```
## Rows: 1,055,293
## Columns: 7
## $ geo_type    <chr> "city", "city", "city", "city", "city"...
## $ region      <chr> "Aachen", "Aachen", "Aachen", "Aachen"...
## $ trans_type   <chr> "driving", "driving", "driving", "driv...
## $ sub_region   <chr> "North Rhine-Westphalia", "North Rhine...
## $ country      <chr> "Germany", "Germany", "Germany", "Germ...
## $ date         <date> 2020-01-13, 2020-01-14, 2020-01-15, 2...
## $ dir_request  <dbl> 100.00, 100.73, 102.86, 102.65, 109.39...
```

Now we can see **date** is formatted correctly

TidyApple: Counting



"data science is mostly counting things" - tabyl vignette

Count the `trans_type` variable with `dplyr::count()`

```
TidyApple %>%  
  count(trans_type)
```

```
## # A tibble: 3 × 2  
##   trans_type     n  
##   <chr>        <int>  
## 1 driving      743682  
## 2 transit       106512  
## 3 walking      205099
```

Add the `sort = TRUE` argument

```
TidyApple %>%  
  count(trans_type, sort = TRUE)
```

```
## # A tibble: 3 × 2  
##   trans_type     n  
##   <chr>        <int>  
## 1 driving      743682  
## 2 walking      205099  
## 3 transit       106512
```

Visualizing Variable Distributions



What kinds of question(s) do graphs like these answer?



What does the distribution of direction requests look like?

What is the distribution of `dir_request`? We will explore this with a histogram.

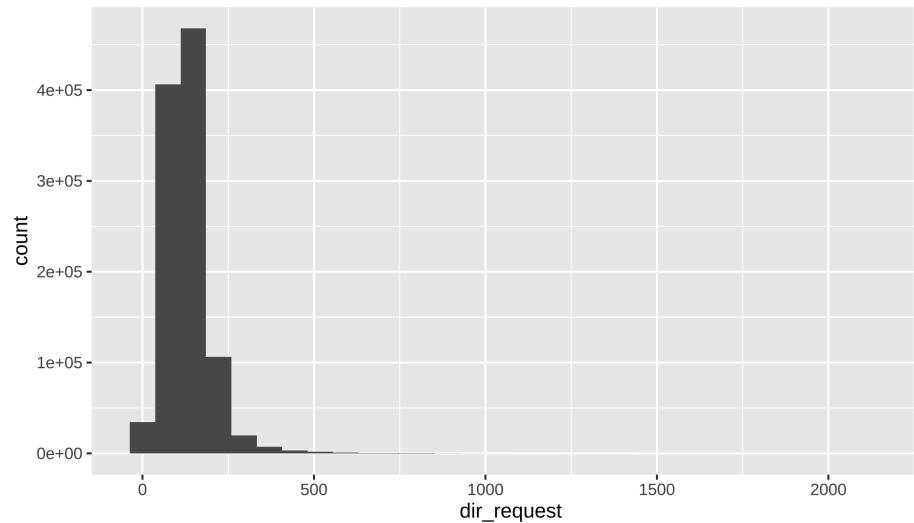
Labels!!

```
lab_hist <- labs(x = "Apple directions requests",
                  y = "Count",
                  title = "Distribution of Direction Requests",
                  subtitle = "source: https://covid19.apple.com/mobility")
```

Histogram = single variable distributions

Create a histogram of `dir_request` with the code below:

```
TidyApple %>% ggplot() +  
  geom_histogram(aes(x = dir_request)) +  
  lab_hist
```



Histograms: Changing the Y Axis



Fix the `y` axis numbers with help from the `scales` package

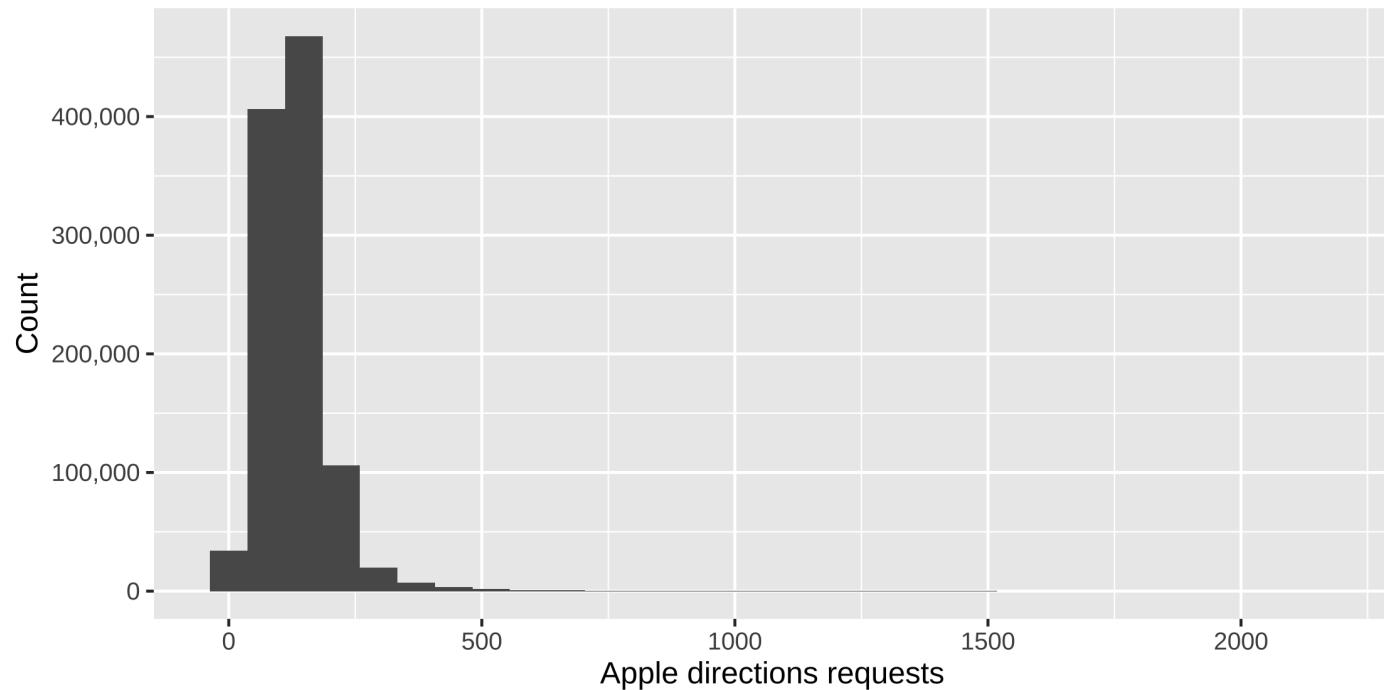
```
library(scales)
TidyApple %>% ggplot() +
  geom_histogram(aes(x = dir_request)) +
  scale_y_continuous(labels = scales::comma) +
  lab_hist
```

Histograms: Changing the Y Axis



Distribution of Direction Requests

source: <https://covid19.apple.com/mobility>

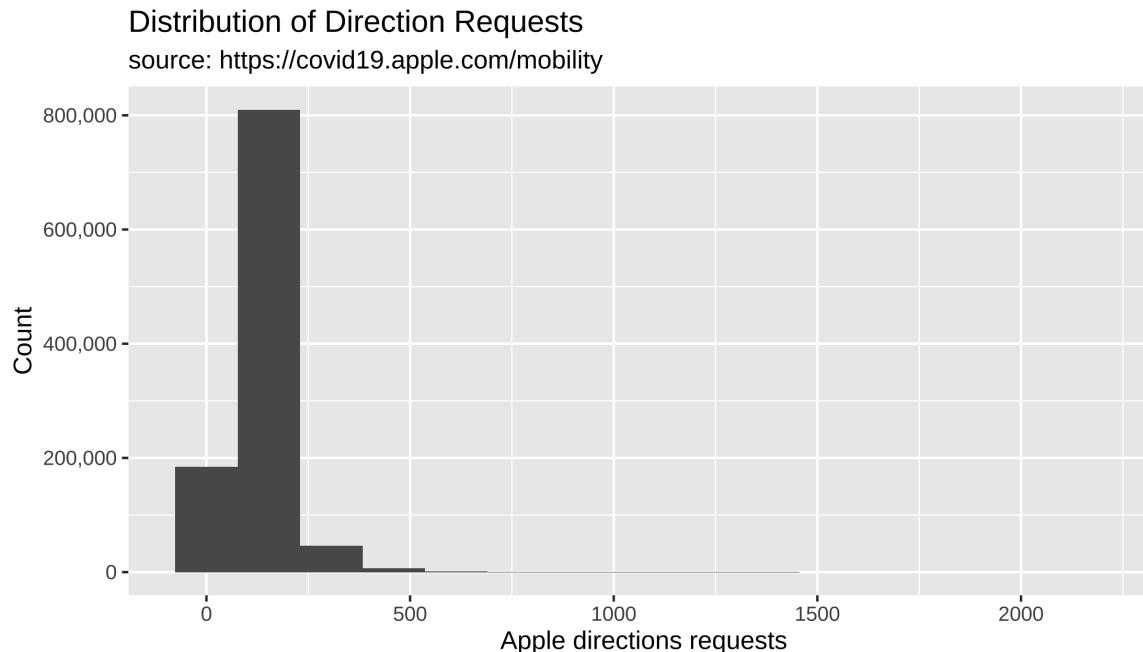


Changing Histogram Shape



Adjust the shape of the histogram with the `bins` argument

```
TidyApple %>% ggplot() +  
  geom_histogram(aes(x = dir_request), bins = 15) +  
  scale_y_continuous(labels = scales::comma) +  
  lab_hist
```

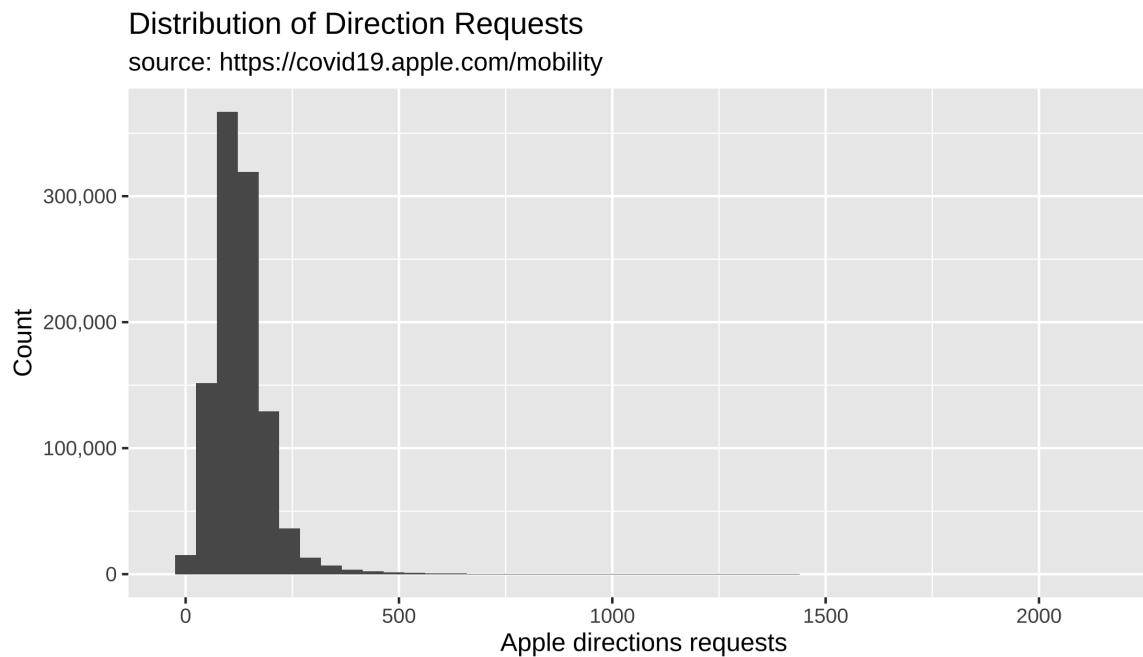


Changing Histogram Shape



Adjust the shape of the histogram with the `bins` argument

```
TidyApple %>% ggplot() +  
  geom_histogram(aes(x = dir_request), bins = 45) +  
  scale_y_continuous(labels = scales::comma) +  
  lab_hist
```



Visualizing variable distributions across groups

What questions do these graphs answer?

How does the distribution of `dir_request` across `trans_type`?

We can view this with a density plot, violin plot, or ridgeline plot



Density Plots



CREATE LABELS FIRST!!

We need a new set of labels

```
lab_density <- labs(x = "Apple directions requests",
                     y = "Density",
                     title = "Direction Requests vs. Transportation Type",
                     subtitle = "source: https://covid19.apple.com/mobility")
```

Density Plots



Visualize the distribution of `dir_request` across `trans_type` with a `geom_density()`

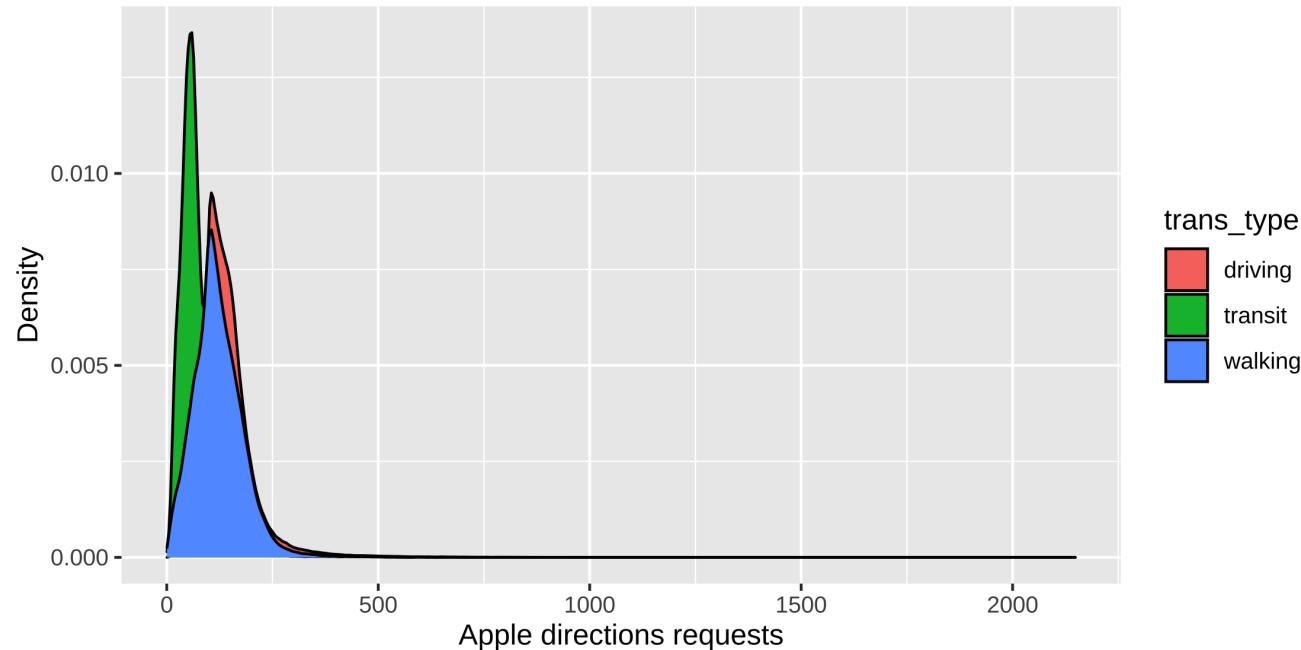
```
TidyApple %>%
  ggplot() +
  geom_density(aes(x = dir_request,
                   fill = trans_type)) +
  lab_density
```

Density Plots



Direction Requests vs. Transportation Type

source: <https://covid19.apple.com/mobility>



Density Plots (alpha)



Adjust the `alpha` so we can see the overlap

```
TidyApple %>%
  ggplot() +
  geom_density(aes(x = dir_request,
                   fill = trans_type),
               alpha = 1/3) +
  lab_density
```

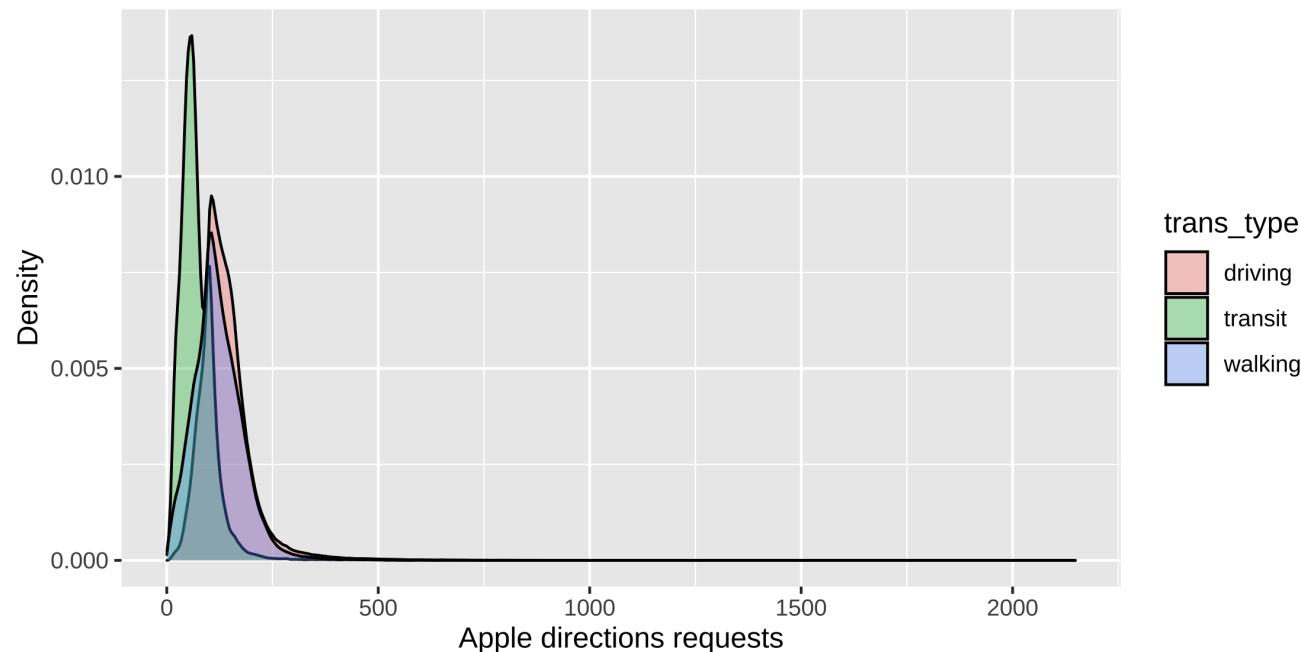
Density Plots (alpha)



Now we can see where they overlap

Direction Requests vs. Transportation Type

source: <https://covid19.apple.com/mobility>



Violin Plots



Violin plots are alternatives to boxplots.

```
lab_violin <- labs(y = "Apple directions requests",
                     x = "Transportation Types",
                     title = "Direction Requests by Transportation Type",
                     subtitle = "source: https://covid19.apple.com/mobility")
```

These allow us to add a categorical variable to the `x` axis.

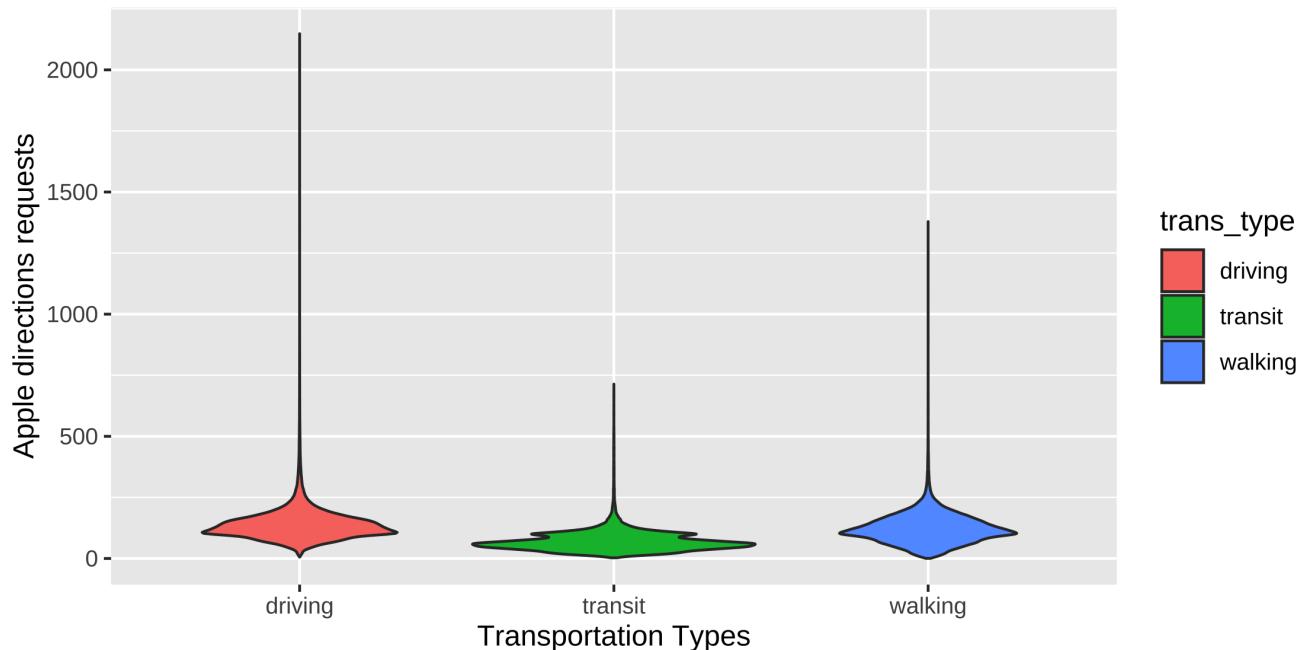
Violin Plots



```
TidyApple %>%
  ggplot() +
  geom_violin(aes(y = dir_request, x = trans_type,
                   fill = trans_type)) +
  lab_violin
```

Direction Requests by Transportation Type

source: <https://covid19.apple.com/mobility>



Violin Plots: confused?



Add a boxplot layer!

```
TidyApple %>%
  ggplot() +
  geom_violin(aes(y = dir_request, x = trans_type,
                  fill = trans_type), alpha = 1/5)
```

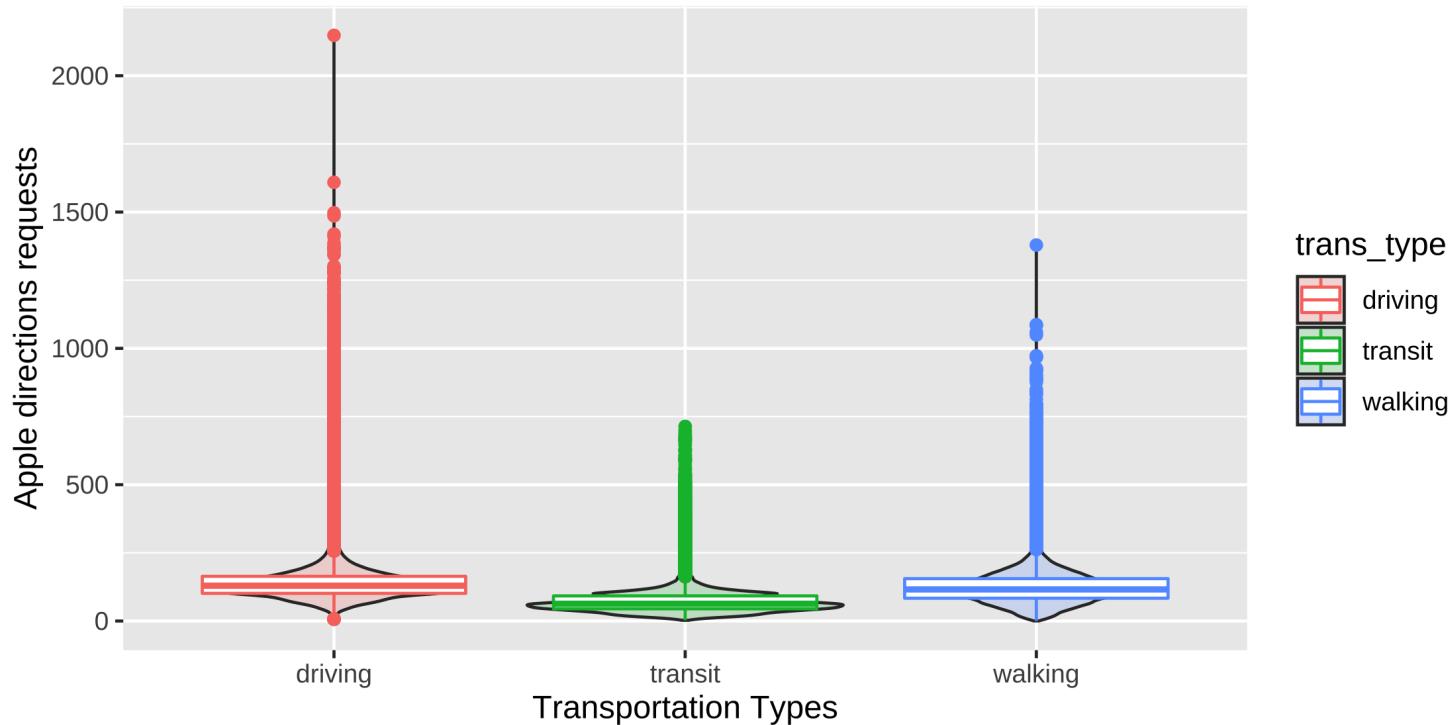
```
TidyApple %>%
  ggplot() +
  geom_violin(aes(y = dir_request, x = trans_type,
                  fill = trans_type), alpha = 1/5) +
  geom_boxplot(aes(y = dir_request, x = trans_type,
                  color = trans_type)) +
  lab_violin
```

Violin Plots and Boxpots



Direction Requests by Transportation Type

source: <https://covid19.apple.com/mobility>



Rideline Plots



Another option is a ridgeline plot (from the `ggridges` package). These display multiple densities.

```
lab_ridges <- labs(x = "Apple directions requests",
                    y = "Transportation Types",
                    title = "Direction Requests by Transportation Type",
                    subtitle = "source: https://covid19.apple.com/mobility")
```

```
library(ggridges)
TidyApple %>%
  ggplot() +
  geom_density_ridges(aes(x = dir_request,
                          y = trans_type,
                          fill = trans_type),
                      alpha = 1/5) +
  lab_ridges
```

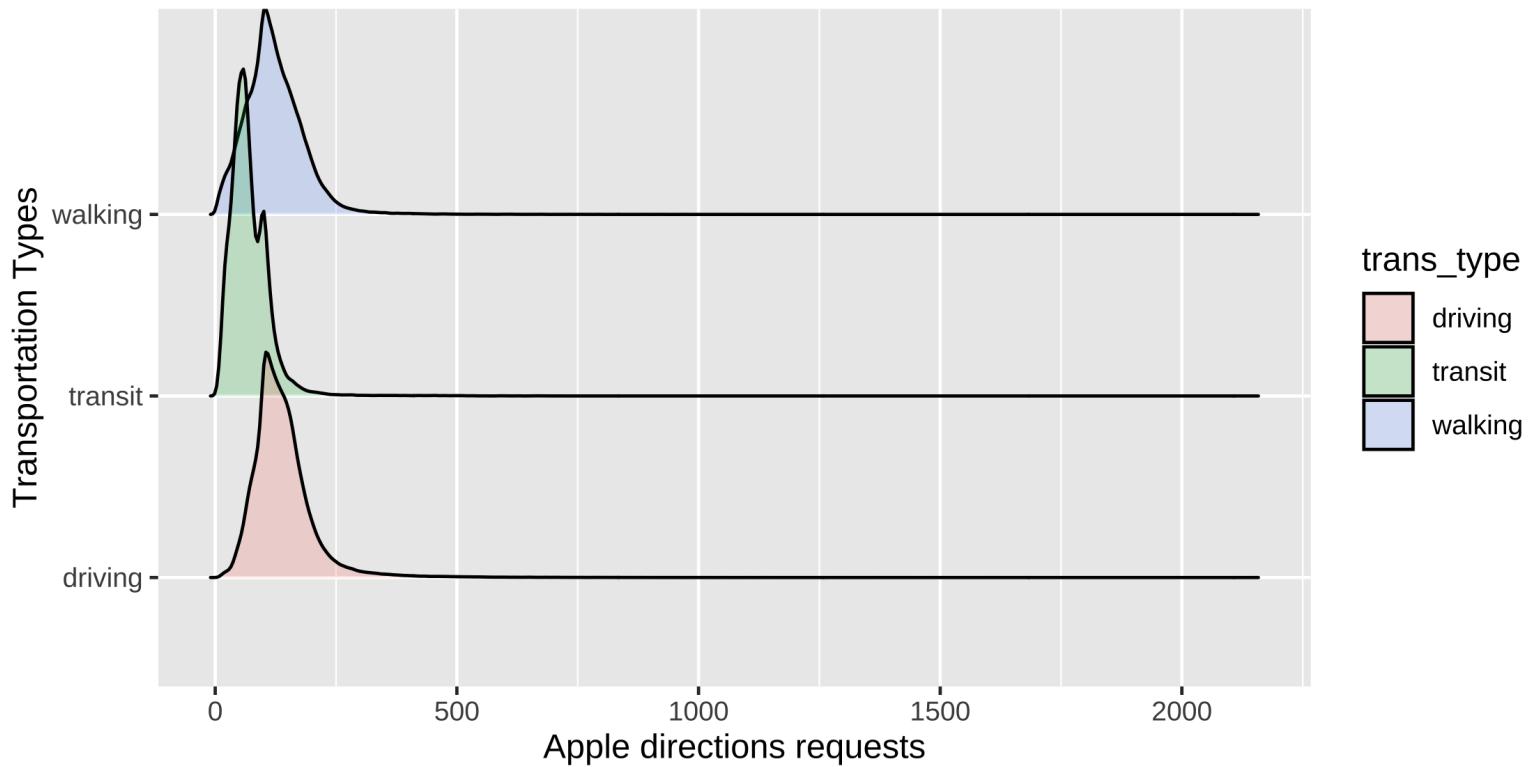
Rideline Plots



We can adjust the alpha on these just like the density plots.

Direction Requests by Transportation Type

source: <https://covid19.apple.com/mobility>



Line Graphs



Narrowing date ranges



Filter the data to only us cities between `2020-02-01` and `2020-08-01`. Use `skimr::skim()` to make sure it works!

```
TidyApple %>%
  # us cities
  filter(geo_type == "city" &
         country == "United States",
         # feb - aug
         date >= lubridate::as_date("2020-02-01") &
         date <= lubridate::as_date("2020-08-01")) %>%
  # check work!
  skimr::skim(date)
```

Check with `skimr` output



These are helpful if we're checking a filter on a numerical certain condition (`min`, `max`, `mean`, etc.)

— Data Summary —————	
	Values
Name	Piped data
Number of rows	57462
Number of columns	7

Column type frequency:	
Date	1

Group variables	None
— Variable type: Date —————	
skim_variable	n_missing complete_rate
1 date	0 1
	min max median n_unique
	2020-02-01 2020-08-01 2020-05-02 183

Narrow to US Cities (Feb-Jul)



Create `USCitiesFebJul` data by filtering to US cities between Feb 1, 2020 and July 31, 2020.

```
TidyApple %>%
  filter(geo_type == "city" &
         country == "United States",
         date >= lubridate::as_date("2020-02-01") &
         date <= lubridate::as_date("2020-08-01")) -> USCitiesFebJul
```

Line Graph: Labels (1)



Ideally, our labels update whenever the data changes. We can do this with `paste0()`.

```
paste0(min(USCitiesFebJul$date),  
       " through ",  
       max(USCitiesFebJul$date))
```

```
## [1] "2020-02-01 through 2020-08-01"
```

```
subtitle = paste0(  
  min(USCitiesFebJul$date), # first date  
  " through ", # plain text  
  max(USCitiesFebJul$date) # last date  
,
```

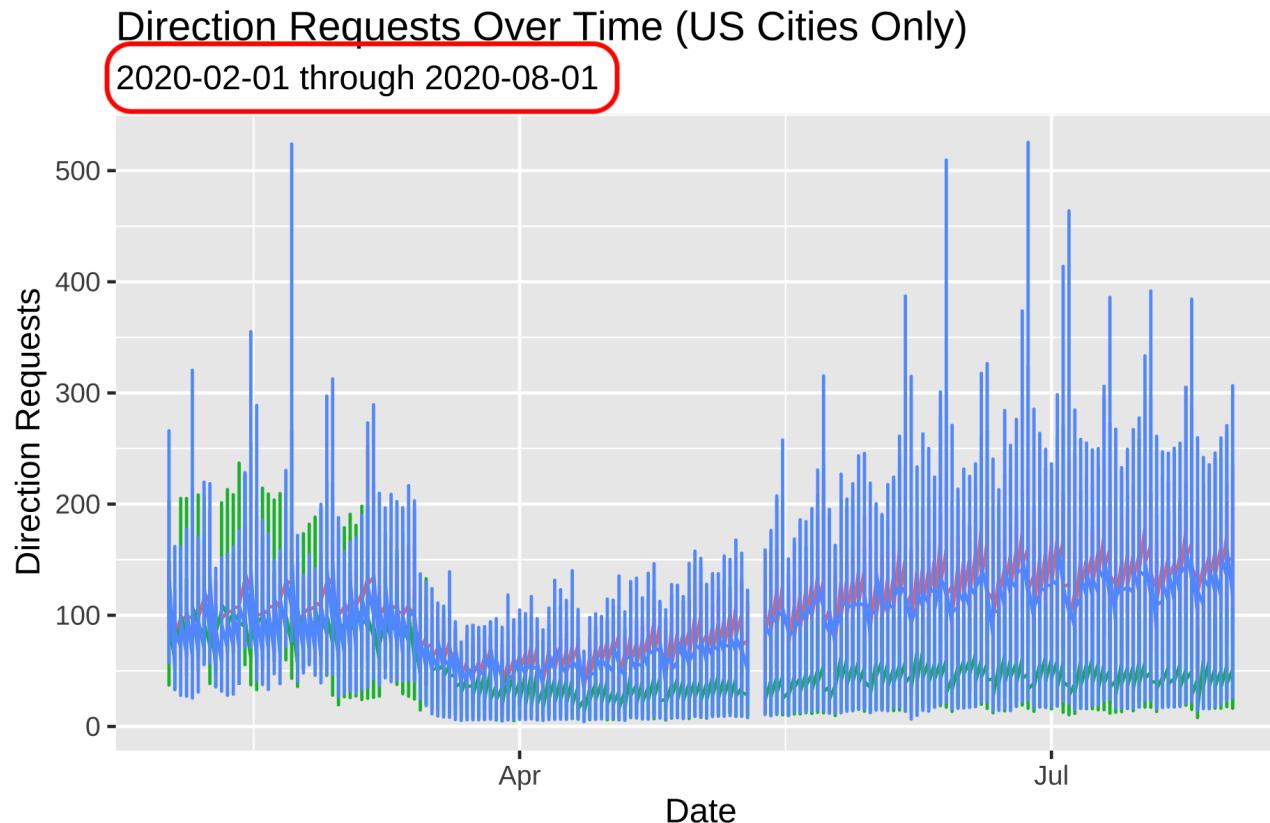
Line Graph: Labels (2)



We can then supply the `subtitle` to `labs`

```
labs(x = "Date",
      y = "Direction Requests",
      title = "Direction Requests Over Time (US Cities Only)",
      subtitle = paste0(min(USCitiesFebJul$date),
                        " through ",
                        max(USCitiesFebJul$date)),
      caption = "source: https://covid19.apple.com/mobility",
      color = "Transit Type") -> lab_line_graph
```

Line Graph: Labels (3)

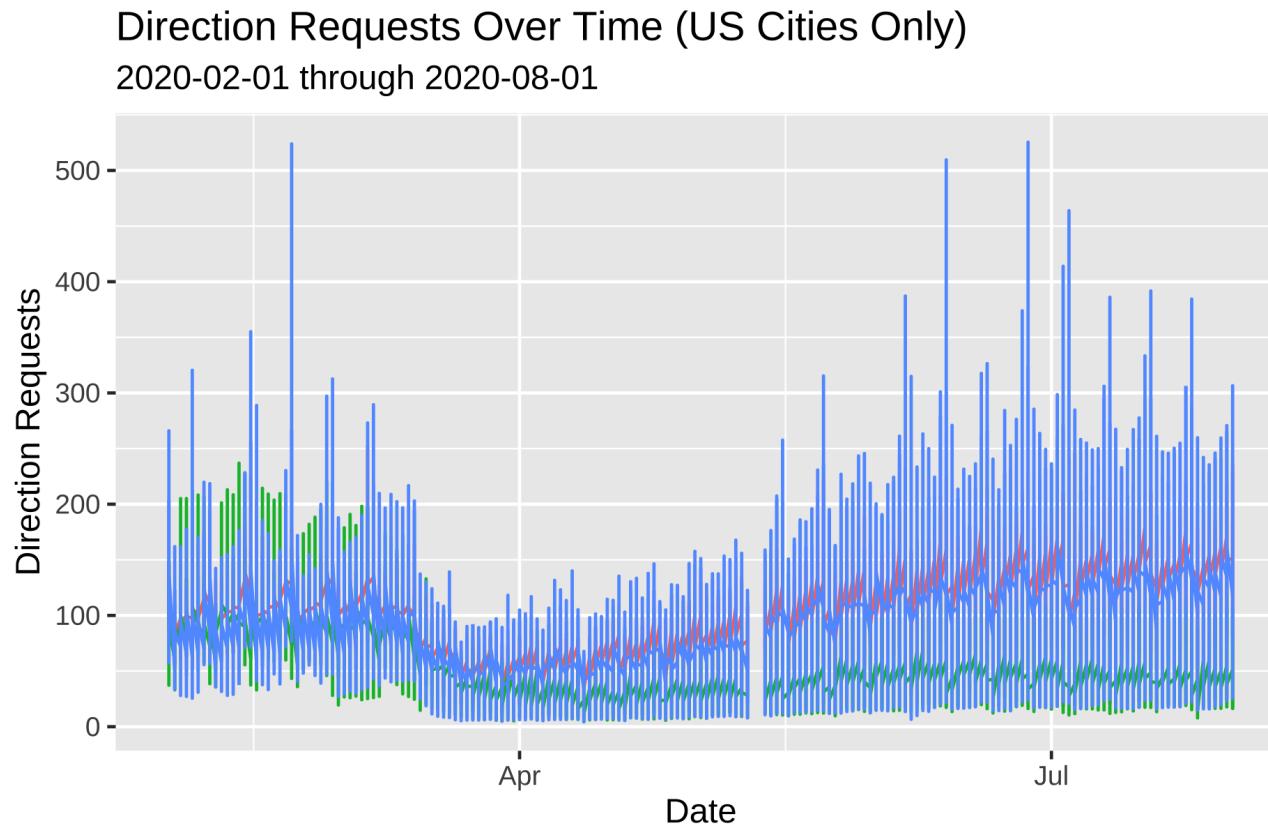


source: <https://covid19.apple.com/mobility>

Line Graphs: Overlapping Lines



Consider our previous line graph--the lines overlap a bit.



source: <https://covid19.apple.com/mobility>

Line Graph: Line Size (1)



We can minimize the size of the line with the `size` aesthetic.

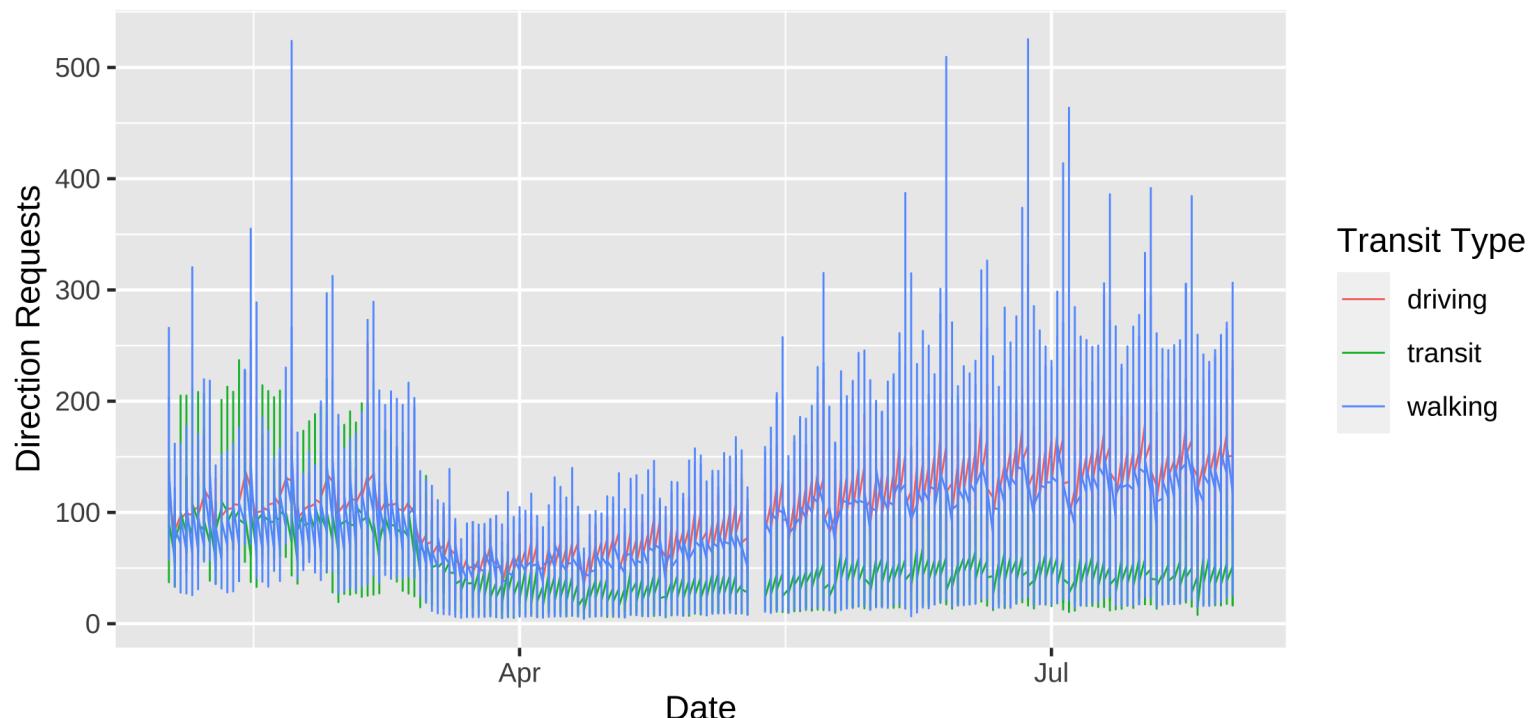
```
USCitiesFebJul %>%
  ggplot() +
  geom_line(aes(x = date, y = dir_request,
                group = trans_type, color = trans_type),
            # make these slightly smaller
            size = 0.30) +
  lab_line_graph
```

Line Graph: Line Size (2)



This makes the trends easier to see.

Direction Requests Over Time (US Cities Only)
2020-02-01 through 2020-08-01



source: <https://covid19.apple.com/mobility>

Adding Text

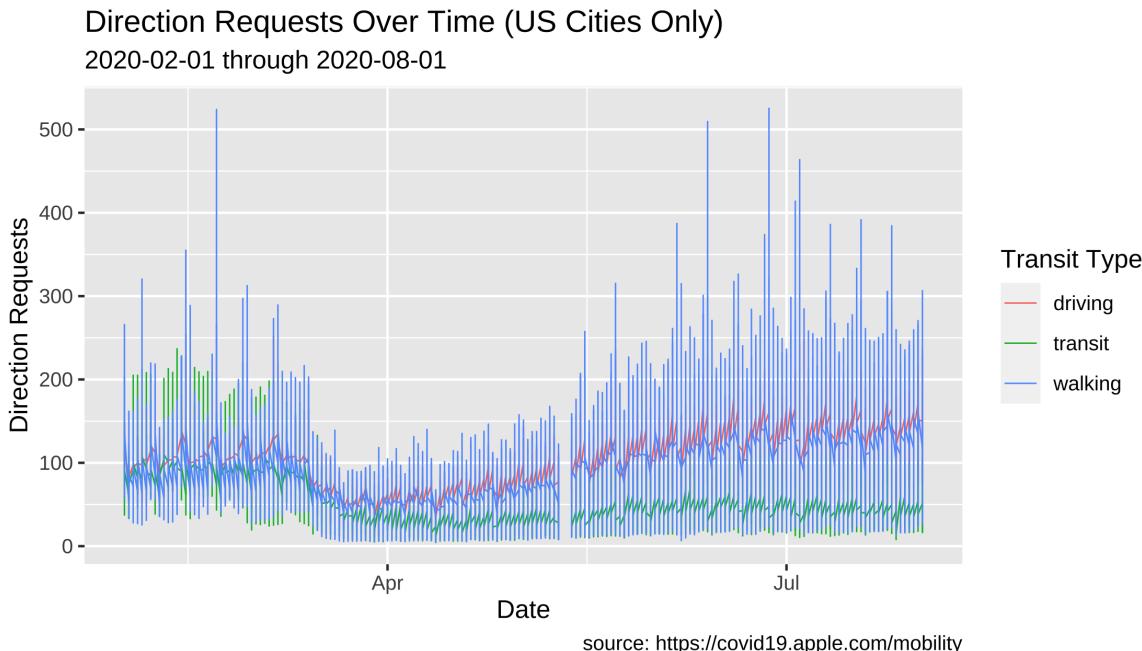


Labeling Missing Data



There is a gap in the direct request data (this is documented in the data source).

"Data for May 11-12 is not available and will appear as blank columns in the data set."



We should annotate this on a line graph!

Create Annotate Data



These data are **filtered** to US cities between March 1, 2020 to June 30, 2020.

```
USCitiesMarJun <- TidyApple %>%
  filter(geo_type == "city" & country == "United States",
         date >= lubridate::as_date("2020-03-01") &
         date <= lubridate::as_date("2020-07-01"))
```

Annotate Data



UScitiesMarJun

geo_type	region	trans_type	sub_region	country	date
<chr>	<chr>	<chr>	<chr>	<chr>	<date>
city	Akron	driving	Ohio	United States	2020-03-01
city	Akron	driving	Ohio	United States	2020-03-02
city	Akron	driving	Ohio	United States	2020-03-03
city	Akron	driving	Ohio	United States	2020-03-04
city	Akron	driving	Ohio	United States	2020-03-05
city	Akron	driving	Ohio	United States	2020-03-06
city	Akron	driving	Ohio	United States	2020-03-07
city	Akron	driving	Ohio	United States	2020-03-08
city	Akron	driving	Ohio	United States	2020-03-09
city	Akron	driving	Ohio	United States	2020-03-10

1-10 of 60 rows | 1-6 of 7 columns

Previous [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [Next](#)

annotate: Build Labels



Build our labels first!

```
labs(x = "Date",
      y = "Direction Requests",
      title = "Spring Direction Requests (Mar-Jun) in US Cities",
      subtitle = paste0(min(USCitiesMarJun$date),
                        " through ",
                        max(USCitiesMarJun$date)),
      caption = "source: https://covid19.apple.com/mobility",
      color = "Transit Type") -> lab_annotation
```

annotate: Build Line Graph



Build a line graph layer (`gg_line_annotate`)

```
USCITIESMarJun %>%
  ggplot() +
  geom_line(aes(x = date, y = dir_request,
                group = trans_type, color = trans_type),
            size = 0.30) -> gg_line_annotate
```

annotate: build coordinate system



Add a coordinate system layer (`gg_coord_system`)

```
coord_cartesian(  
  xlim = c(min(USCitiesMarJun$date),  
           max(USCitiesMarJun$date)),  
  
  ylim = c(min(USCitiesMarJun$dir_request, na.rm = TRUE),  
           max(USCitiesMarJun$dir_request, na.rm = TRUE))) ->  
  gg_coord_system
```

annotate: build line segment



Build vertical line segment (`gg_line_segment`)

```
annotate(geom = "segment",
         size = 1,
         color = "firebrick3",
         x = lubridate::as_date("2020-05-11"),
         xend = lubridate::as_date("2020-05-11"),
         y = 270,
         yend = 100) -> gg_line_segment
```

annotate: Build Text Annotation



Build text annotation (`gg_text_annotation`)

```
annotate(geom = "text",
         color = "red",
         hjust = 0.5,
         size = 6,
         x = lubridate::as_date("2020-05-07"),
         y = 300,
         label = "Data not available") -> gg_text_annotation
```

annotate: Combine Layers



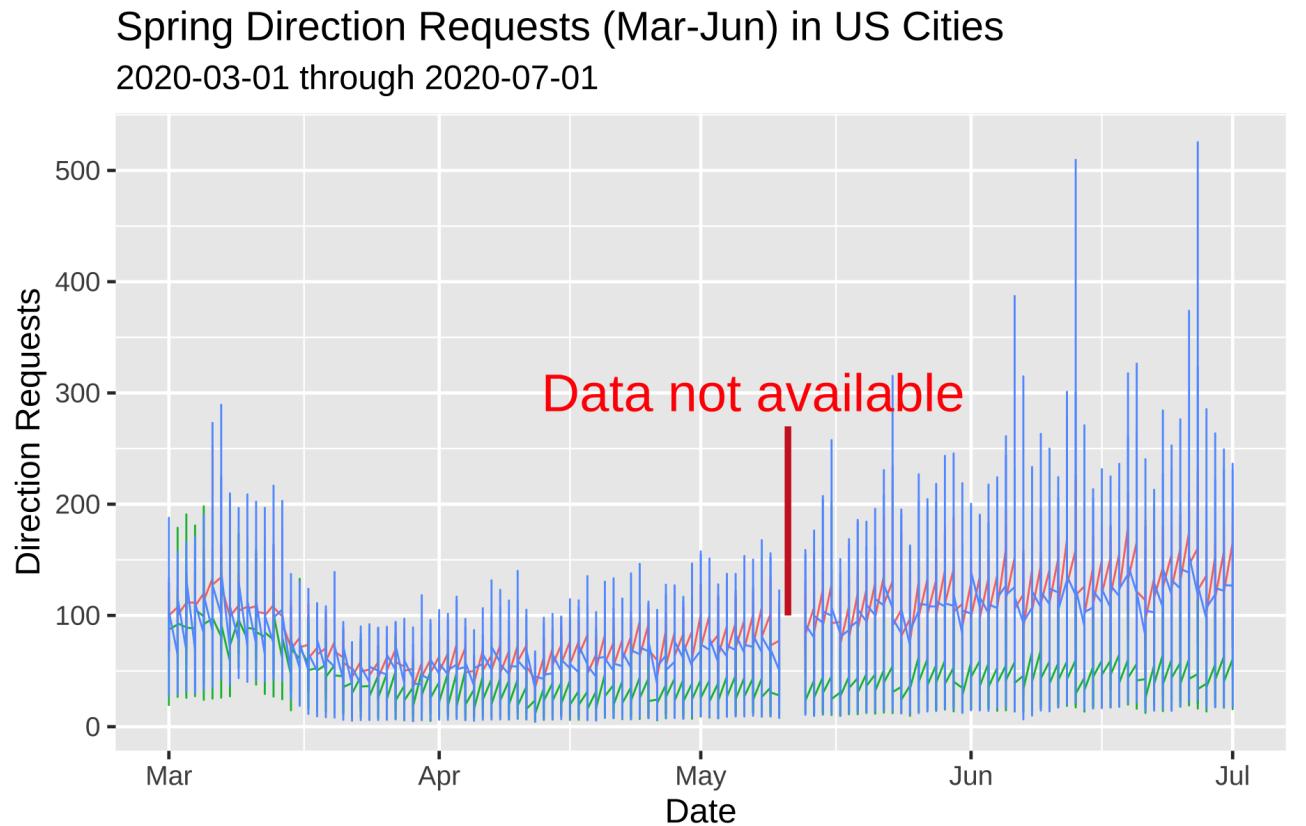
Now we can use the `ggplot2` syntax to combine these layers...

```
gg_line_annotate + # line graph
  gg_coord_system + # coordinate system
  gg_line_segment + # line annotation
  gg_text_annotation + # text annotation
  lab_annotation # labels
```

annotate: Complete Graph



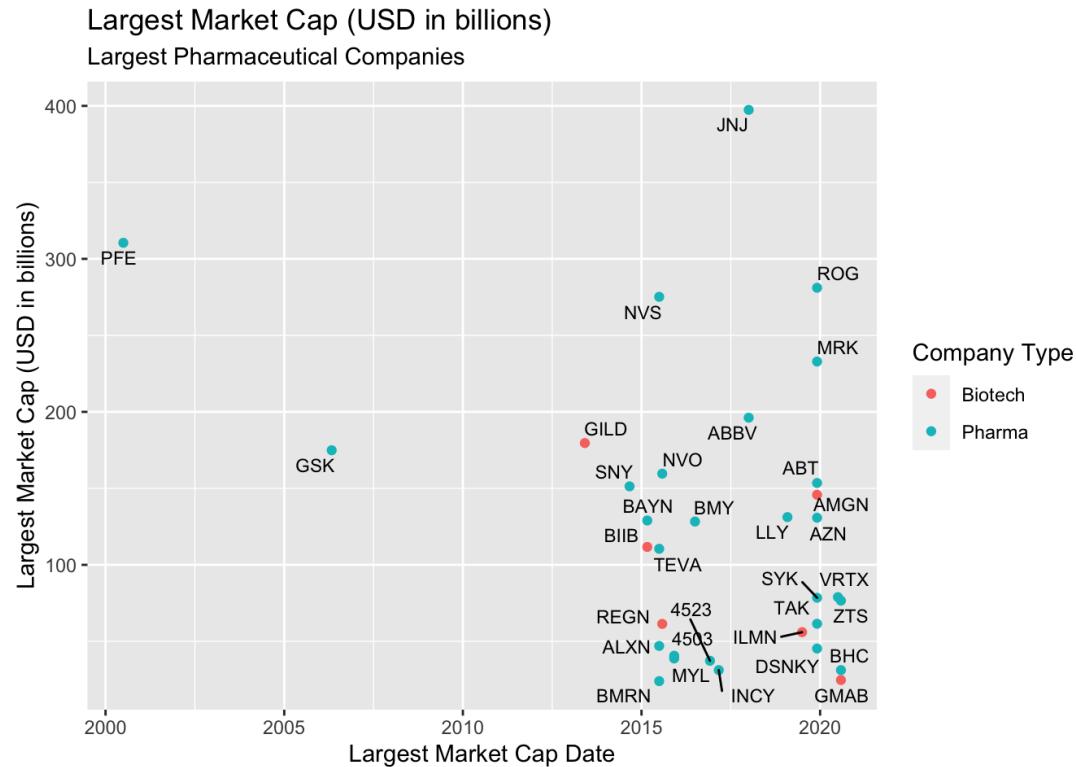
...and we see a complete graph!



Labeling Values (Review)



In the previous slides, we learned about labeling values with `ggrepel`.



Now we're going to extend this to plotting text and values on our graphs!

Highlighting Large US Cities



Filter TidyApple to the 5 largest US cities (by population).

```
TopUSCities <- TidyApple %>%
  filter(country == "United States" &
         region %in% c("New York City", "Los Angeles",
                       "Chicago", "Houston", "Phoenix"))
```

View TopUSCities



TopUSCities

geo_type	region	trans_type	sub_region	country	date
<chr>	<chr>	<chr>	<chr>	<chr>	<date>
city	Chicago	driving	Illinois	United States	2020-01-13
city	Chicago	driving	Illinois	United States	2020-01-14
city	Chicago	driving	Illinois	United States	2020-01-15
city	Chicago	driving	Illinois	United States	2020-01-16
city	Chicago	driving	Illinois	United States	2020-01-17
city	Chicago	driving	Illinois	United States	2020-01-18
city	Chicago	driving	Illinois	United States	2020-01-19
city	Chicago	driving	Illinois	United States	2020-01-20
city	Chicago	driving	Illinois	United States	2020-01-21
city	Chicago	driving	Illinois	United States	2020-01-22

1-10 of 60 rows | 1-6 of 7 columns

Previous [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [Next](#)

Highlighting Peak Driving



Create a dataset with only the maximum direction request values for "driving" per `region`.

```
TopUSCities %>%
  filter(trans_type == "driving") %>%
  group_by(region) %>%
  slice_max(dir_request) %>%
  ungroup() -> MaxUSCitiesDriving
```

View MaxUSCitiesDriving



MaxUSCitiesDriving

geo_type	region	trans_type	sub_region	country
<chr>	<chr>	<chr>	<chr>	<chr>
city	Chicago	driving	Illinois	United States
city	Houston	driving	Texas	United States
city	Los Angeles	driving	California	United States
city	New York City	driving	New York	United States
city	Phoenix	driving	Arizona	United States

5 rows | 1-5 of 7 columns

Create graph labels



We know we want to see the max direction requests labeled, so we will update the labels for the graph.

```
lab_line_max_drivers <- labs(  
  x = "Date",  
  y = "Direction Requests",  
  title = "Peak Driving Direction Requests in Largest US Cities",  
  subtitle = paste0(min(TopUSCities$date),  
    " through ",  
    max(TopUSCities$date)),  
  caption = "source: https://covid19.apple.com/mobility",  
  color = "Transit Type")
```

Create Value Labels



We will also use `paste0()` here to create a variable for the labels that combines the city and date.

```
MaxUSCitiesDriving <- MaxUSCitiesDriving %>%  
  mutate(max_driving_labels = paste0(region, ", ", date))
```

View Value Labels



Take a look at the labels we've created

```
MaxUSCitiesDriving %>%  
  select(max_driving_labels)
```

max_driving_labels

<chr>

Chicago, 2020-07-17

Houston, 2020-02-14

Los Angeles, 2020-02-14

New York City, 2020-09-04

Phoenix, 2020-02-29

5 rows

Create Line Layer + Label Layer

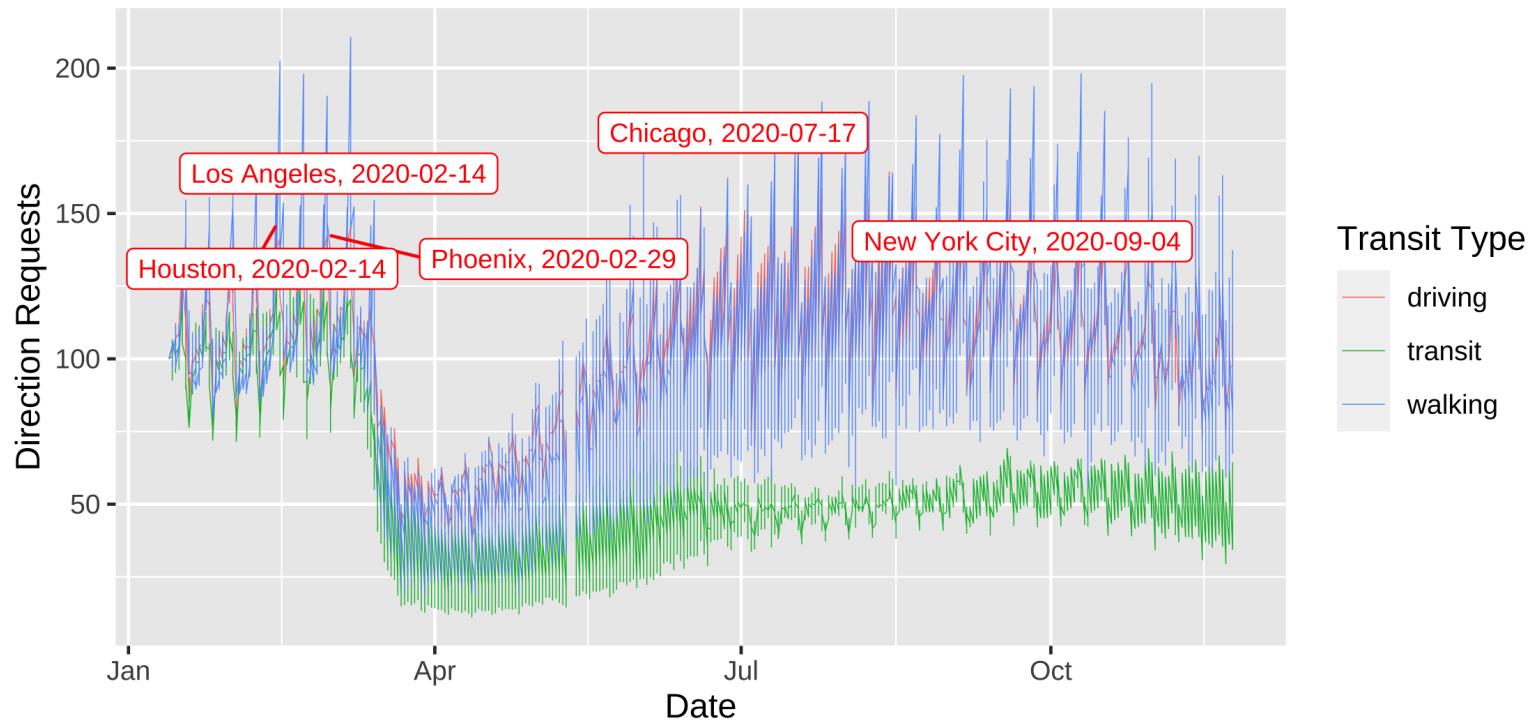


```
library(ggrepel)
TopUSCities %>%
  ggplot() +
  geom_line(aes(x = date, y = dir_request,
                group = trans_type,
                color = trans_type),
            # make these slightly smaller again...
            size = 0.15) +
  geom_label_repel(data = MaxUSCitiesDriving,
                   aes(x = date, y = dir_request,
                       label = max_driving_labels),
                   # set color and size...
                   color = "red",
                   size = 3) +
  lab_line_max_drivers
```

View plot with labels



Peak Driving Direction Requests in Largest US Cities
2020-01-13 through 2020-11-24



Adding Reference Lines



Reference Line Data



TopUSCities

geo_type	region	trans_type	sub_region	country	date
<chr>	<chr>	<chr>	<chr>	<chr>	<date>
city	Chicago	driving	Illinois	United States	2020-01-13
city	Chicago	driving	Illinois	United States	2020-01-14
city	Chicago	driving	Illinois	United States	2020-01-15
city	Chicago	driving	Illinois	United States	2020-01-16
city	Chicago	driving	Illinois	United States	2020-01-17
city	Chicago	driving	Illinois	United States	2020-01-18
city	Chicago	driving	Illinois	United States	2020-01-19
city	Chicago	driving	Illinois	United States	2020-01-20
city	Chicago	driving	Illinois	United States	2020-01-21
city	Chicago	driving	Illinois	United States	2020-01-22

1-10 of 60 rows | 1-6 of 7 columns

Previous [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [Next](#)

Reference Line Labels



```
labs(x = "Date",
     y = "Direction Requests",
     title = "Trends of Relative Activity in Selected US Cities",
     subtitle = "New York, Los Angeles, Chicago, Houston, Phoenix",
     color = "Type") -> lab_ref_lines
```

Horizontal Reference Lines



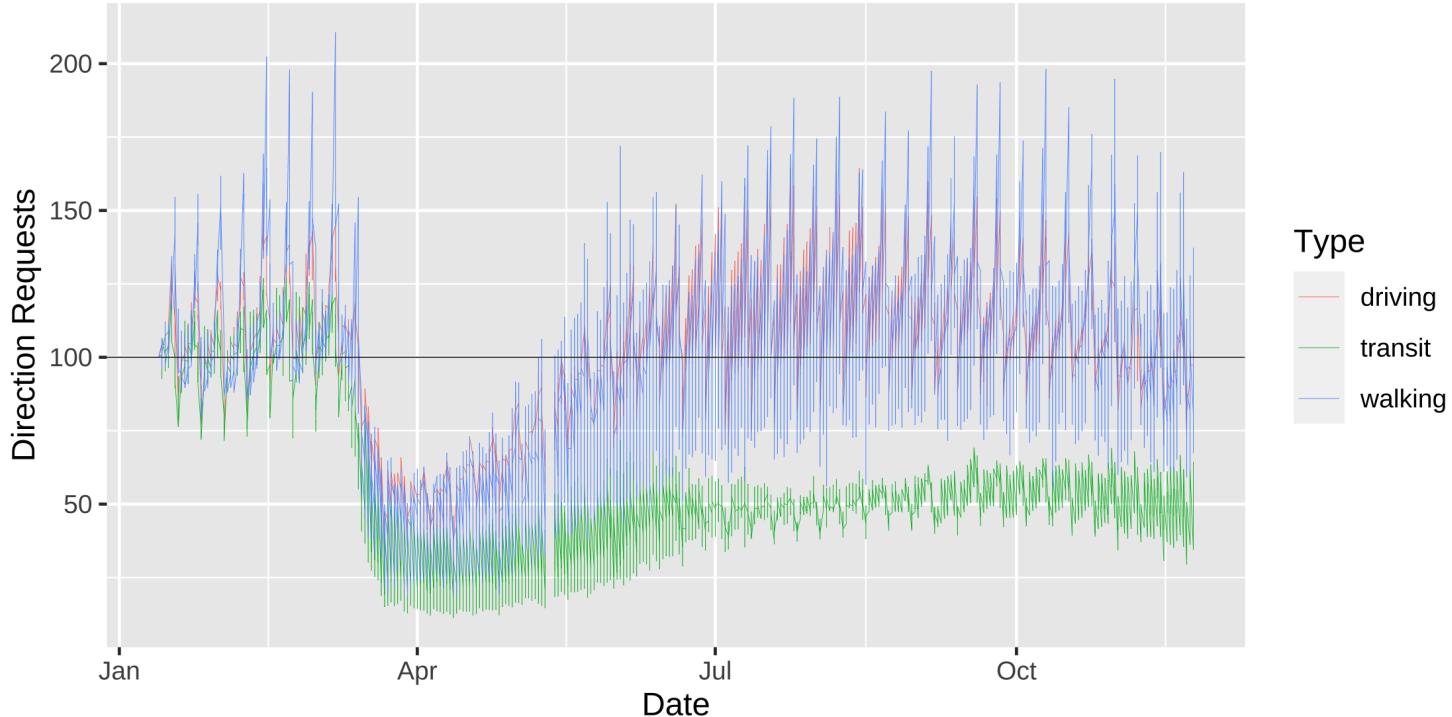
Add the `geom_hline()` for a horizontal reference line (at 100)

```
TopUSCities %>%
  ggplot(aes(x = date, y = dir_request,
             group = trans_type,
             color = trans_type)) +
  geom_line(size = 0.1) +
  geom_hline(yintercept = 100, size = 0.2, color = "gray20") +
  lab_ref_lines
```

Horizontal Reference Lines



Trends of Relative Activity in Selected US Cities
New York, Los Angeles, Chicago, Houston, Phoenix



Advanced Facetting



Faceting basics



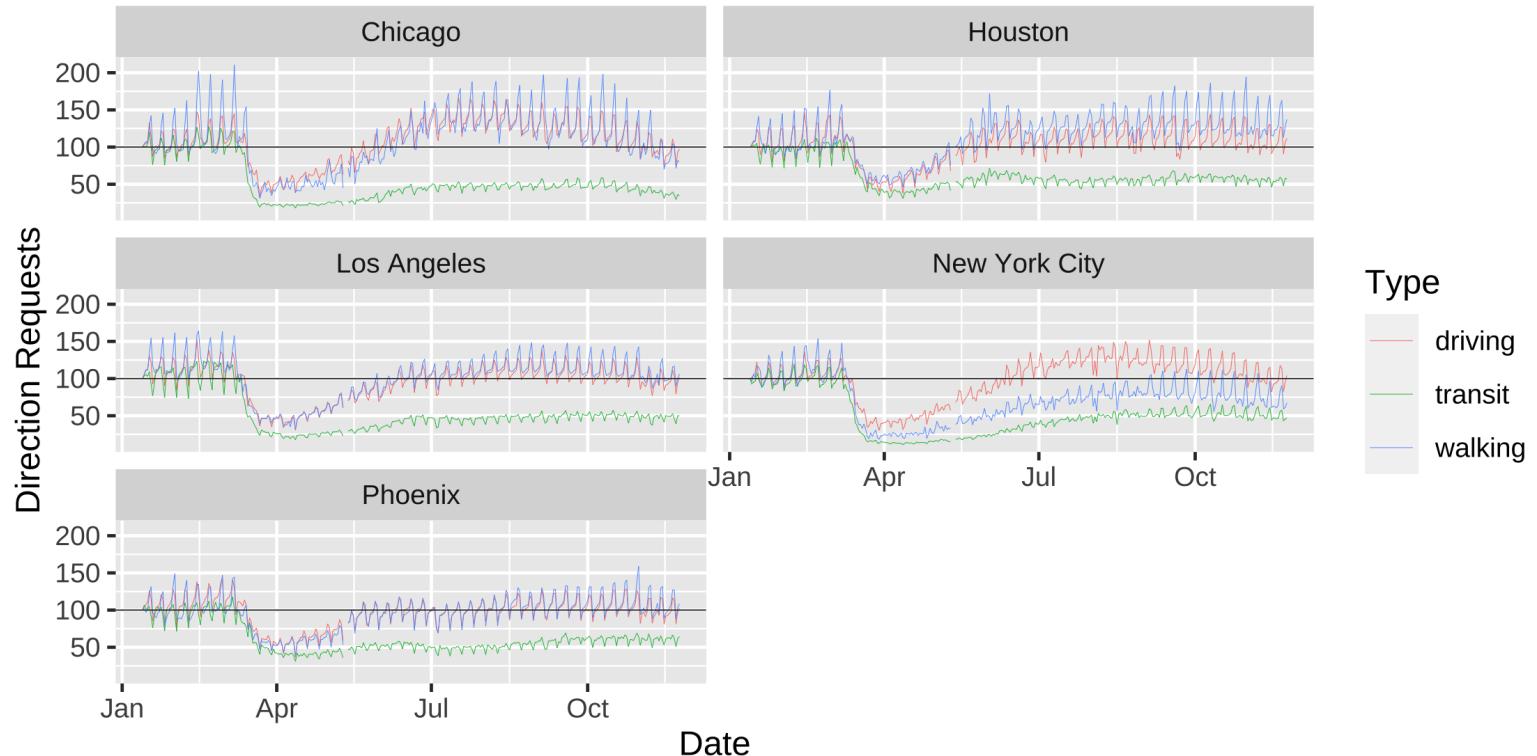
Facets create subplots across levels of a categorical variable.

```
TopUSCities %>%
  ggplot(aes(x = date, y = dir_request,
             group = trans_type,
             color = trans_type)) +
  geom_line(size = 0.1) +
  geom_hline(yintercept = 100,
             size = 0.2,
             color = "gray20") +
  facet_wrap(~ region, ncol = 2) +
  lab_ref_lines
```

Single Variable Facets



Trends of Relative Activity in Selected US Cities
New York, Los Angeles, Chicago, Houston, Phoenix



Multiple Variable Facets

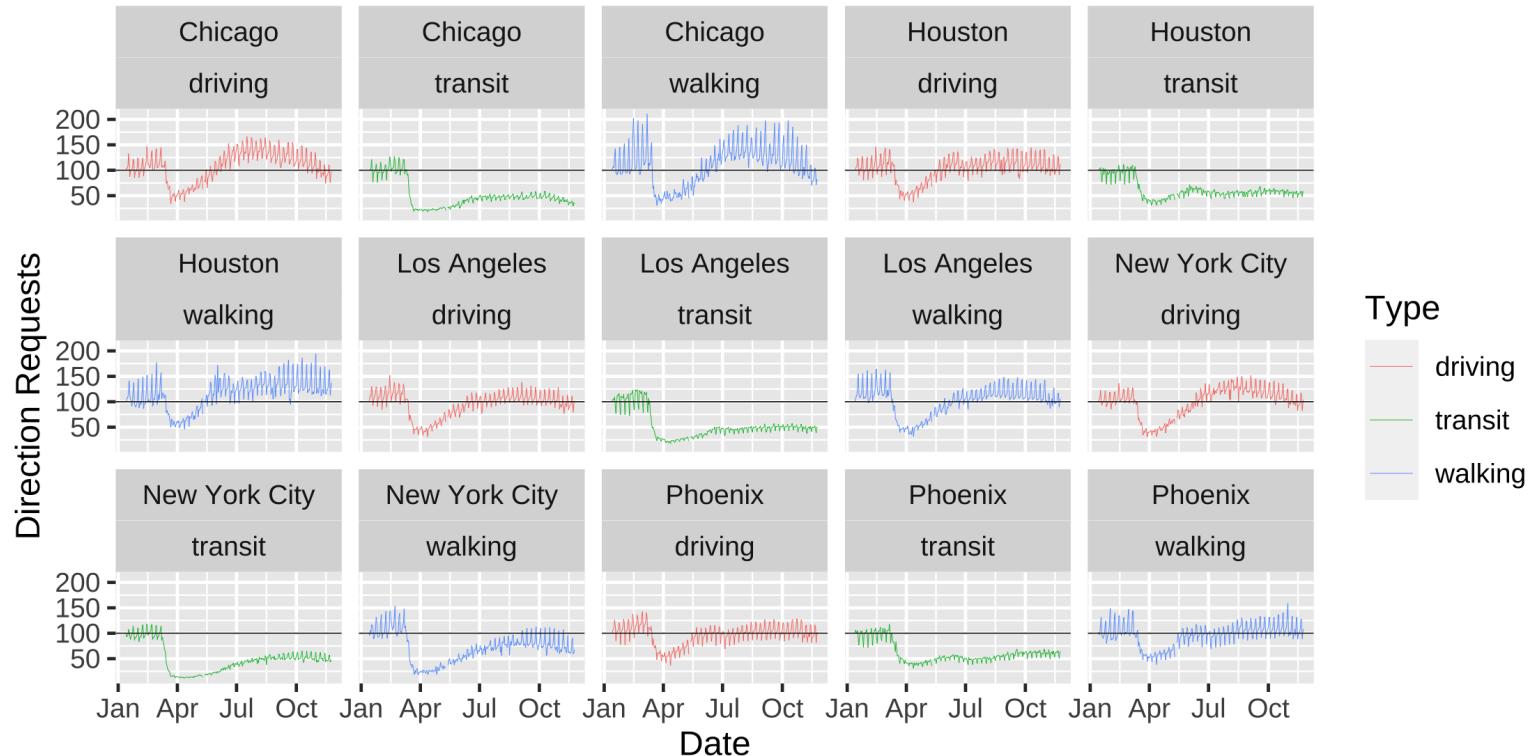


```
TopUSCities %>%
  ggplot(aes(x = date, y = dir_request,
             group = region,
             color = trans_type)) +
  geom_line(size = 0.1) +
  geom_hline(yintercept = 100,
             size = 0.2,
             color = "gray20") +
  facet_wrap(region ~ trans_type, ncol = 5) +
  lab_ref_lines
```

Multiple Variable Facets



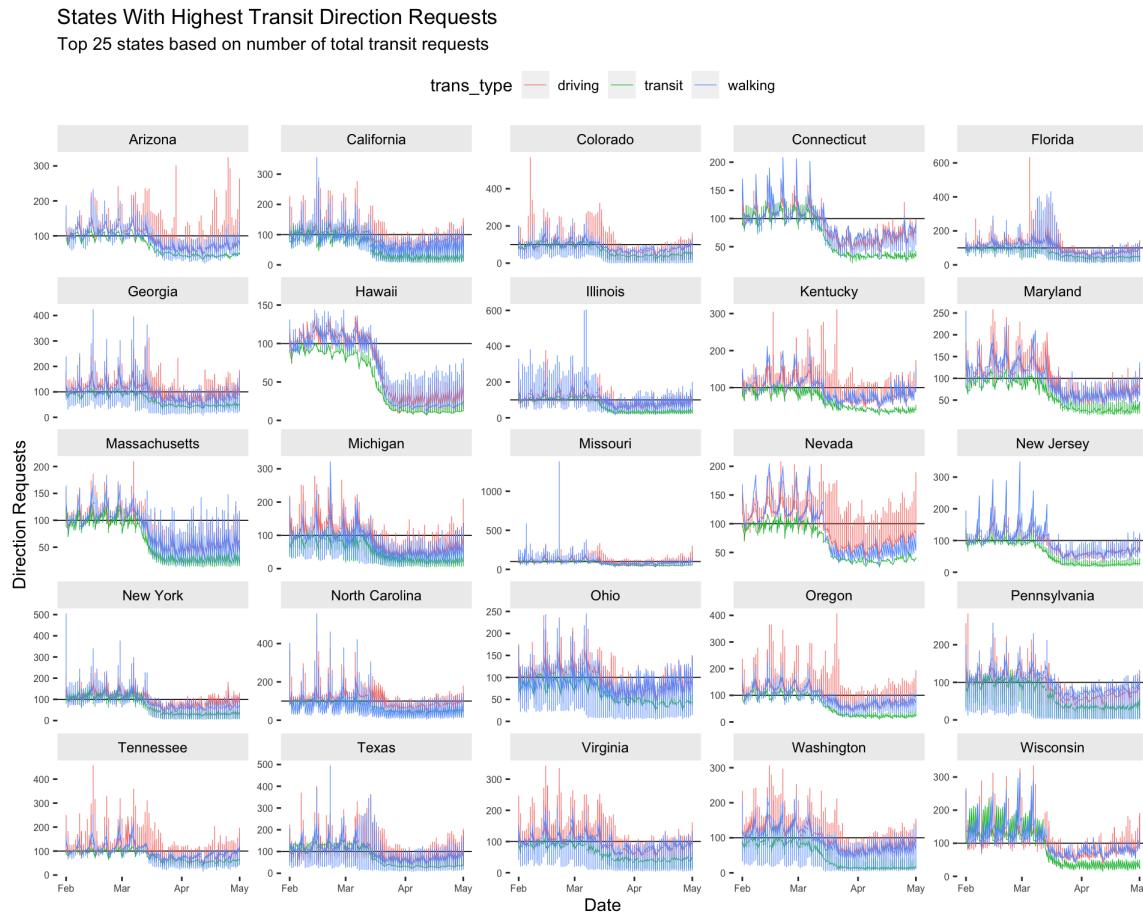
Trends of Relative Activity in Selected US Cities
New York, Los Angeles, Chicago, Houston, Phoenix



Advanced Faceting (`facet_wrap_paginate`)



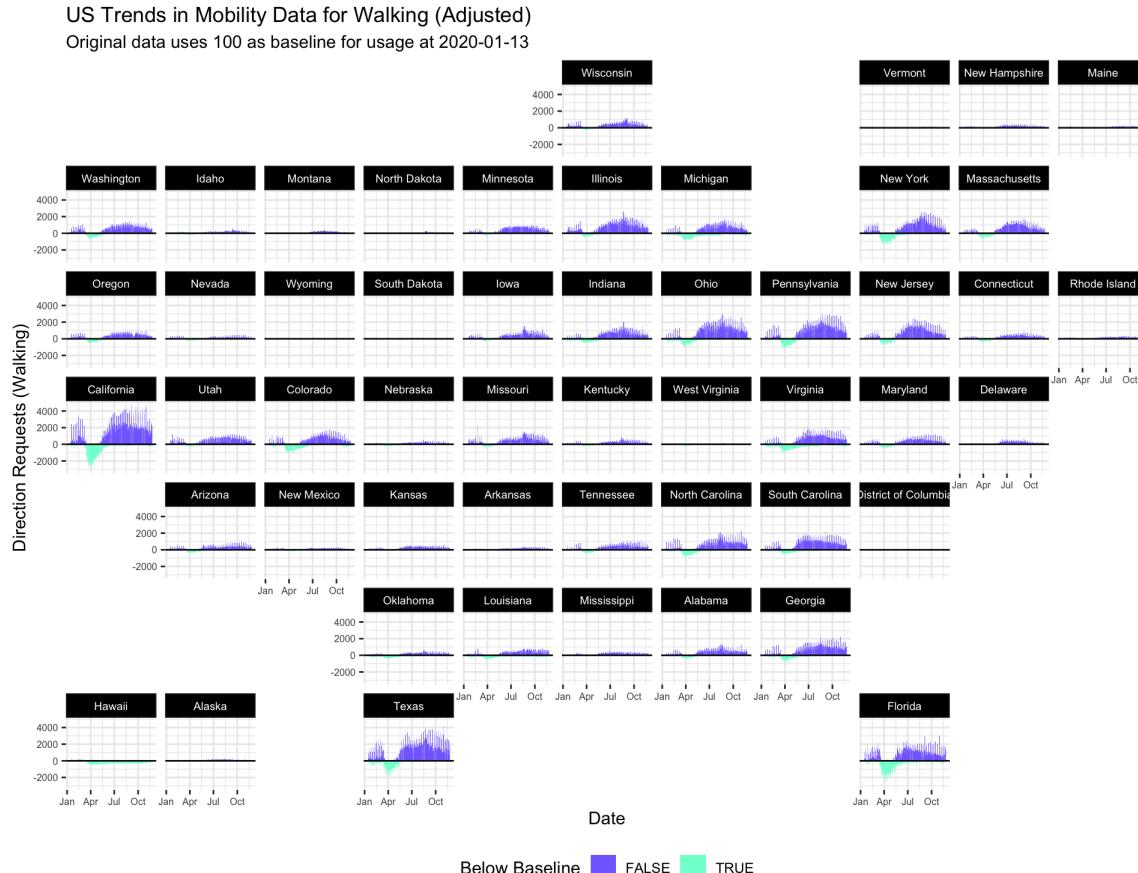
Check out the exercises for more advanced faceting with `facet_wrap_paginate()` from the `ggforce` package.



Advanced Faceting (`facet_geo`)



Check out the exercises for more advanced faceting with `facet_geo()` from the `geofacet` package.



More Resources



Fundamentals of Data Visualization

ggplot2 extensions gallery

R Graphics Cookbook